题目要求：
本次学习的语法是选择语句和循环语句，需要注意的是本次使用的语法做了一些改进，不是纯粹的python2语法。

需要结合上次课四则运算的解析程序

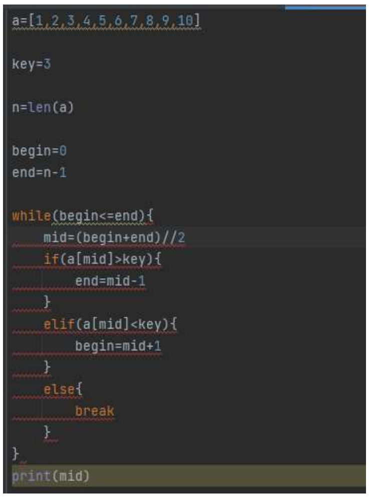
（1）示例程序位于example3/

（2）需要进行解析的文件为binary_search.py和select_sort.py，分别对应二分查找和选择排序。

（3）需要完成以下内容的解析
- if
- while
- for

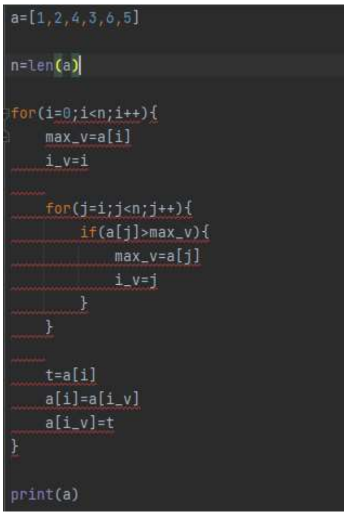
（4）解析结果以语法树的形式呈现

binary_search.py文件内容如下：



```
a=[1,2,3,4,5,6,7,8,9,10]

key=3

n=len(a)

begin=0
end=n-1

while(begin<=end){
    mid=(begin+end)//2
    if(a[mid]>key){
        end=mid-1
    }
    elif(a[mid]<key){
        begin=mid+1
    }
    else{
        break
    }
}
print(mid)
```

select_sort.py文件内容如下：

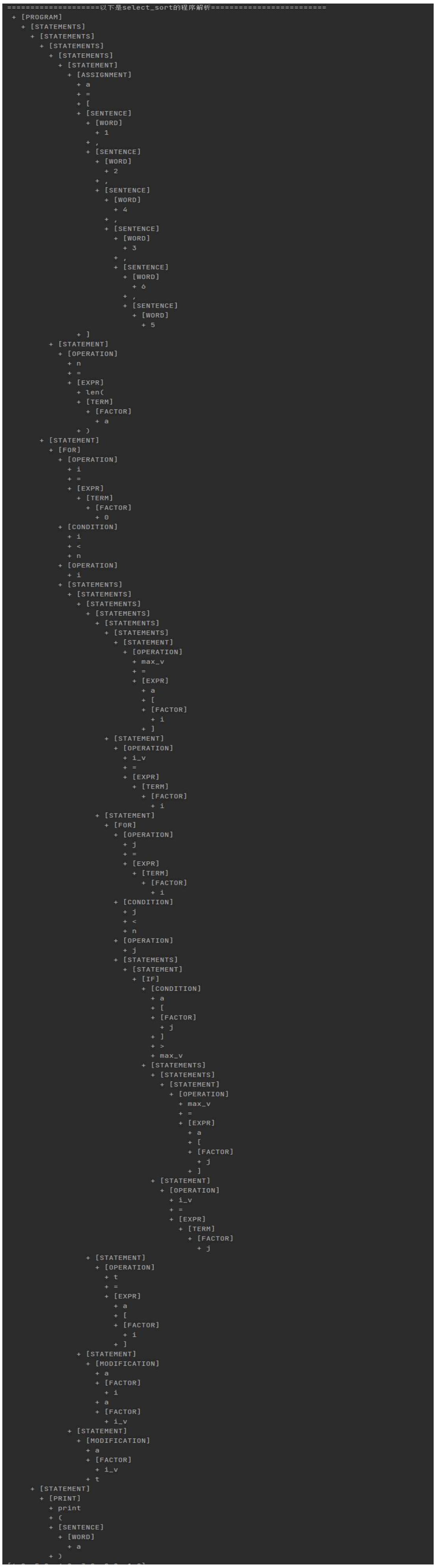


```
a=[1,2,4,3,6,5]

n=len(a)

for(i=0;i<n;i++){
    max_v=a[i]
    i_v=i

    for(j=i;j<n;j++){
        if(a[j]>max_v){
            max_v=a[j]
            i_v=j
        }
    }

    t=a[i]
    a[i]=a[i_v]
    a[i_v]=t
}

print(a)
```

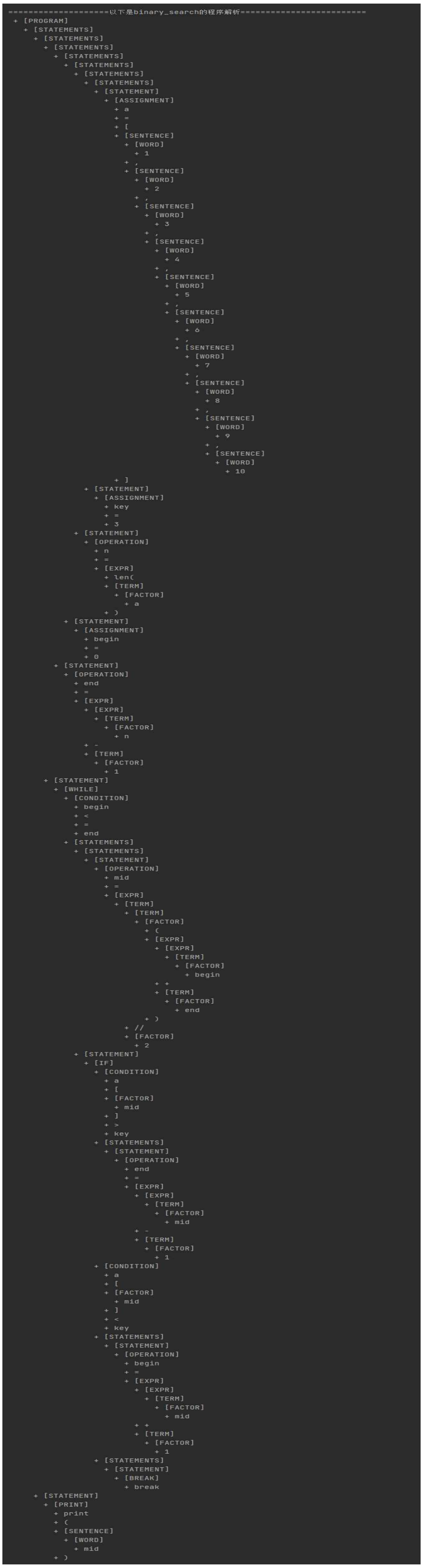
程序说明：

1. 打开 main.py 文件，确保 source 中的所有代码在同一目录下
2. 确保已经安装了 PLY 库
3. 运行 main.py 文件
4. 对 binary_search.py 和 select_sort.py 文件中的程序段进行解析，结果以语法树的形式展现，并展示 print 的结果以及所有变量的最终值字典，解析结果如下图：

   第一张图是 select_sort.py 的运行结果，第二张是 binary_search.py 的运行结果



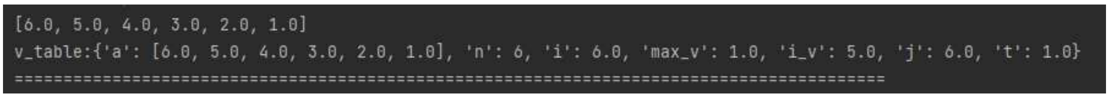
```
====================以下是select_sort的程序解析=========================
 + [PROGRAM]
   + [STATEMENTS]
     + [STATEMENTS]
       + [STATEMENTS]
         + [STATEMENTS]
           + [STATEMENT]
             + [ASSIGNMENT]
               + a
               + =
               + [
               + [SENTENCE]
                 + [WORD]
                   + 1
                 + ,
                 + [SENTENCE]
                   + [WORD]
                     + 2
                   + ,
                   + [SENTENCE]
                     + [WORD]
                       + 4
                     + ,
                     + [SENTENCE]
                       + [WORD]
                         + 3
                       + ,
                       + [SENTENCE]
                         + [WORD]
                           + 6
                         + ,
                         + [SENTENCE]
                           + [WORD]
                             + 5
               + ]
         + [STATEMENT]
           + [OPERATION]
             + n
             + =
             + [EXPR]
               + len(
               + [TERM]
                 + [FACTOR]
                   + a
               + )
       + [STATEMENT]
         + [FOR]
           + [OPERATION]
             + i
             + =
             + [EXPR]
               + [TERM]
                 + [FACTOR]
                   + 0
           + [CONDITION]
             + i
             + <
             + n
           + [OPERATION]
             + i
           + [STATEMENTS]
             + [STATEMENTS]
               + [STATEMENTS]
                 + [STATEMENTS]
                   + [STATEMENTS]
                     + [STATEMENTS]
                       + [STATEMENT]
                         + [OPERATION]
                           + max_v
                           + =
                           + [EXPR]
                             + a
                             + [
                             + [FACTOR]
                               + i
                             + ]
                     + [STATEMENT]
                       + [OPERATION]
                         + i_v
                         + =
                         + [EXPR]
                           + [TERM]
                             + [FACTOR]
                               + i
                   + [STATEMENT]
                     + [FOR]
                       + [OPERATION]
                         + j
                         + =
                         + [EXPR]
                           + [TERM]
                             + [FACTOR]
                               + i
                       + [CONDITION]
                         + j
                         + <
                         + n
                       + [OPERATION]
                         + j
                       + [STATEMENTS]
                         + [STATEMENT]
                           + [IF]
                             + [CONDITION]
                               + a
                               + [
                               + [FACTOR]
                                 + j
                               + ]
                               + >
                               + max_v
                             + [STATEMENTS]
                               + [STATEMENTS]
                                 + [STATEMENT]
                                   + [OPERATION]
                                     + max_v
                                     + =
                                     + [EXPR]
                                       + a
                                       + [
                                       + [FACTOR]
                                         + j
                                       + ]
                               + [STATEMENT]
                                 + [OPERATION]
                                   + i_v
                                   + =
                                   + [EXPR]
                                     + [TERM]
                                       + [FACTOR]
                                         + j
                 + [STATEMENT]
                   + [OPERATION]
                     + t
                     + =
                     + [EXPR]
                       + a
                       + [
                       + [FACTOR]
                         + i
                       + ]
               + [STATEMENT]
                 + [MODIFICATION]
                   + a
                   + [FACTOR]
                     + i
                   + a
                   + [FACTOR]
                     + i_v
             + [STATEMENT]
               + [MODIFICATION]
                 + a
                 + [FACTOR]
                   + i_v
                 + t
     + [STATEMENT]
       + [PRINT]
         + print
         + (
         + [SENTENCE]
           + [WORD]
             + a
         + )
```



```
====================以下是binary_search的程序解析=========================
 + [PROGRAM]
   + [STATEMENTS]
     + [STATEMENTS]
       + [STATEMENTS]
         + [STATEMENTS]
           + [STATEMENTS]
             + [STATEMENTS]
               + [STATEMENTS]
                 + [STATEMENT]
                   + [ASSIGNMENT]
                     + a
                     + =
                     + [
                     + [SENTENCE]
                       + [WORD]
                         + 1
                       + ,
                       + [SENTENCE]
                         + [WORD]
                           + 2
                         + ,
                         + [SENTENCE]
                           + [WORD]
                             + 3
                           + ,
                           + [SENTENCE]
                             + [WORD]
                               + 4
                             + ,
                             + [SENTENCE]
                               + [WORD]
                                 + 5
                               + ,
                               + [SENTENCE]
                                 + [WORD]
                                   + 6
                                 + ,
                                 + [SENTENCE]
                                   + [WORD]
                                     + 7
                                   + ,
                                   + [SENTENCE]
                                     + [WORD]
                                       + 8
                                     + ,
                                     + [SENTENCE]
                                       + [WORD]
                                         + 9
                                       + ,
                                       + [SENTENCE]
                                         + [WORD]
                                           + 10
                     + ]
               + [STATEMENT]
                 + [ASSIGNMENT]
                   + key
                   + =
                   + 3
             + [STATEMENT]
               + [OPERATION]
                 + n
                 + =
                 + [EXPR]
                   + len(
                   + [TERM]
                     + [FACTOR]
                       + a
                   + )
           + [STATEMENT]
             + [ASSIGNMENT]
               + begin
               + =
               + 0
         + [STATEMENT]
           + [OPERATION]
             + end
             + =
             + [EXPR]
               + [EXPR]
                 + [TERM]
                   + [FACTOR]
                     + n
               + -
               + [TERM]
                 + [FACTOR]
                   + 1
       + [STATEMENT]
         + [WHILE]
           + [CONDITION]
             + begin
             + <
             + =
             + end
           + [STATEMENTS]
             + [STATEMENTS]
               + [STATEMENT]
                 + [OPERATION]
                   + mid
                   + =
                   + [EXPR]
                     + [TERM]
                       + [TERM]
                         + [FACTOR]
                           + (
                           + [EXPR]
                             + [EXPR]
                               + [TERM]
                                 + [FACTOR]
                                   + begin
                             + +
                             + [TERM]
                               + [FACTOR]
                                 + end
                           + )
                       + //
                       + [FACTOR]
                         + 2
             + [STATEMENT]
               + [IF]
                 + [CONDITION]
                   + a
                   + [
                   + [FACTOR]
                     + mid
                   + ]
                   + >
                   + key
                 + [STATEMENTS]
                   + [STATEMENT]
                     + [OPERATION]
                       + end
                       + =
                       + [EXPR]
                         + [EXPR]
                           + [TERM]
                             + [FACTOR]
                               + mid
                         + -
                         + [TERM]
                           + [FACTOR]
                             + 1
                 + [CONDITION]
                   + a
                   + [
                   + [FACTOR]
                     + mid
                   + ]
                   + <
                   + key
                 + [STATEMENTS]
                   + [STATEMENT]
                     + [OPERATION]
                       + begin
                       + =
                       + [EXPR]
                         + [EXPR]
                           + [TERM]
                             + [FACTOR]
                               + mid
                         + +
                         + [TERM]
                           + [FACTOR]
                             + 1
                 + [STATEMENTS]
                   + [STATEMENT]
                     + [BREAK]
                       + break
     + [STATEMENT]
       + [PRINT]
         + print
         + (
         + [SENTENCE]
           + [WORD]
             + mid
         + )
```

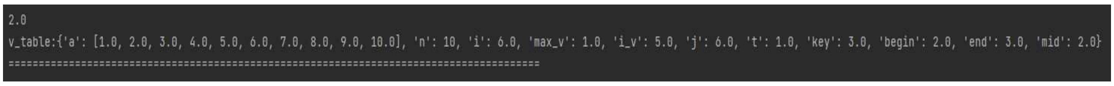
两个程序的 print 结果以及对应的最后 v_table 内容如下：

选择排序的结果：



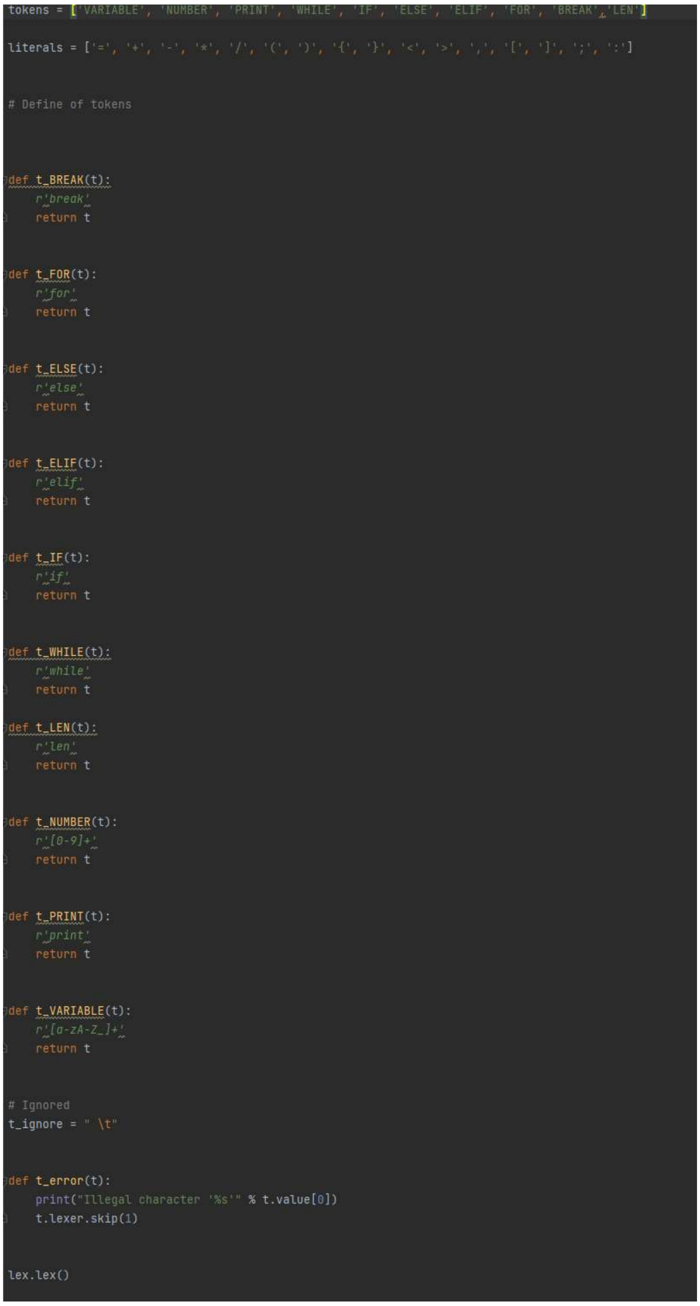
```
[6.0, 5.0, 4.0, 3.0, 2.0, 1.0]
v_table:{'a': [6.0, 5.0, 4.0, 3.0, 2.0, 1.0], 'n': 6, 'i': 6.0, 'max_v': 1.0, 'i_v': 5.0, 'j': 6.0, 't': 1.0}
==========================================================================================
```

二分查找的结果：



```
2.0
v_table:{'a': [1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0], 'n': 10, 'i': 6.0, 'max_v': 1.0, 'i_v': 5.0, 'j': 6.0, 't': 1.0, 'key': 3.0, 'begin': 2.0, 'end': 3.0, 'mid': 2.0}
==========================================================================================
```

5. 对 Lexer 程序定义的 token 规则的解释



```
tokens = ['VARIABLE', 'NUMBER', 'PRINT', 'WHILE', 'IF', 'ELSE', 'ELIF', 'FOR', 'BREAK','LEN']

literals = ['=', '+', '-', '*', '/', '(', ')', '{', '}', '<', '>', ',', '[', ']', ';', ':']


# Define of tokens



def t_BREAK(t):
    r'break'
    return t


def t_FOR(t):
    r'for'
    return t


def t_ELSE(t):
    r'else'
    return t


def t_ELIF(t):
    r'elif'
    return t


def t_IF(t):
    r'if'
    return t


def t_WHILE(t):
    r'while'
    return t

def t_LEN(t):
    r'len'
    return t


def t_NUMBER(t):
    r'[0-9]+'
    return t


def t_PRINT(t):
    r'print'
    return t


def t_VARIABLE(t):
    r'[a-zA-Z_]+'
    return t


# Ignored
t_ignore = " \t"


def t_error(t):
    print("Illegal character '%s'" % t.value[0])
    t.lexer.skip(1)


lex.lex()
```

定义的 token 对应到待解析语句中的一些关键字如 len()，while，for 等的匹配和使用，literals 中定义了需要使用的标点符号。

6. Yacc 语法规则的设计

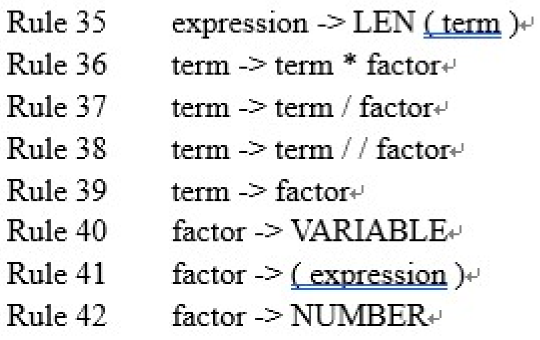
设计的语法规则展开后如下所示：

```
Grammar
Rule 0     S' -> program
Rule 1     program -> statements
Rule 2     statements -> statements statement
Rule 3     statements -> statement
Rule 4     statement -> assignment
Rule 5     statement -> operation
Rule 6     statement -> print
Rule 7     statement -> modification
Rule 8     statement -> iF
Rule 9     statement -> whilE
Rule 10    statement -> for
Rule 11    statement -> break
Rule 12    break -> BREAK statements
Rule 13    break -> BREAK
Rule 14    for -> FOR ( operation ; condition ; operation ) { statements }
Rule 15    condition -> VARIABLE > VARIABLE
Rule 16    condition -> VARIABLE < VARIABLE
Rule 17    condition -> VARIABLE > NUMBER
Rule 18    condition -> VARIABLE < NUMBER
Rule 19    condition -> VARIABLE < = VARIABLE
Rule 20    condition -> VARIABLE [ factor ] > VARIABLE
Rule 21    condition -> VARIABLE [ factor ] < VARIABLE
Rule 22    iF -> IF ( condition ) { statements }
Rule 23    iF -> IF ( condition ) { statements } ELIF ( condition ) { statements }
ELSE { statements }
Rule 24    whilE -> WHILE ( condition ) { statements }
Rule 25    assignment -> VARIABLE = NUMBER
Rule 26    assignment -> VARIABLE = [ sentence ]
Rule 27    modification -> VARIABLE [ factor ] = VARIABLE [ factor ]
Rule 28    modification -> VARIABLE [ factor ] = VARIABLE
Rule 29    operation -> VARIABLE = expression
Rule 30    operation -> VARIABLE + +
Rule 31    expression -> expression + term
Rule 32    expression -> expression - term
```

Rule 33	expression -> term
Rule 34	expression -> VARIABLE [factor]
Rule 35	expression -> LEN (term)
Rule 36	term -> term * factor
Rule 37	term -> term / factor
Rule 38	term -> term / / factor
Rule 39	term -> factor
Rule 40	factor -> VARIABLE
Rule 41	factor -> (expression)
Rule 42	factor -> NUMBER
Rule 43	print -> PRINT (sentence)
Rule 44	sentence -> word , sentence
Rule 45	sentence -> word
Rule 46	word -> NUMBER
Rule 47	word -> VARIABLE

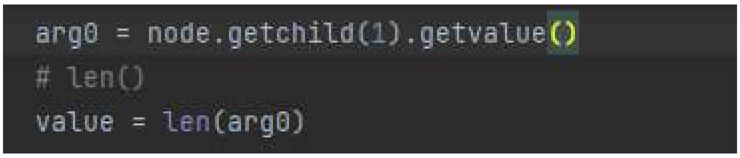
7. Translation 的关键部分逻辑设计

（1） len()函数的求值和赋值



Rule 35	expression -> LEN (term)↵
Rule 36	term -> term * factor↵
Rule 37	term -> term / factor↵
Rule 38	term -> term / / factor↵
Rule 39	term -> factor↵
Rule 40	factor -> VARIABLE↵
Rule 41	factor -> (expression)↵
Rule 42	factor -> NUMBER↵

求长度的部分规则在这，可以发现 term 最终可以归约到某一个变量或者数字上去。

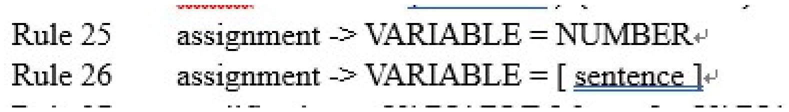


```
arg0 = node.getchild(1).getvalue()
# len()
value = len(arg0)
```

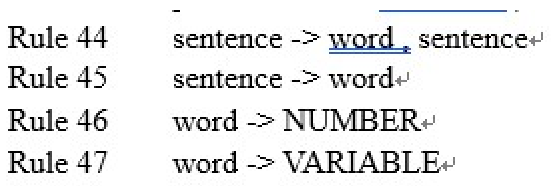
对孩子结点取值计算长度就行了

（2） 对变量进行数组类型赋值

规则如下

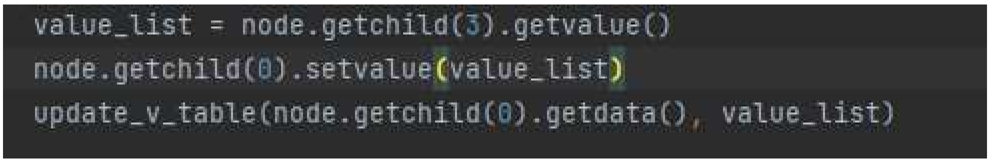


Rule 25	assignment -> VARIABLE = NUMBER↵
Rule 26	assignment -> VARIABLE = [sentence]↵



Rule 44	sentence -> word , sentence↵
Rule 45	sentence -> word↵
Rule 46	word -> NUMBER↵
Rule 47	word -> VARIABLE↵

对应到 translation 的代码:

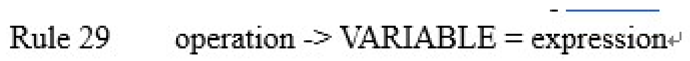


```
value_list = node.getchild(3).getvalue()
node.getchild(0).setvalue(value_list)
update_v_table(node.getchild(0).getdata(), value_list)
```

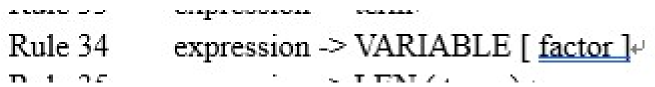
先把 list 从子结点的值中取出来，然后再赋给变量，更新 v_table

（3） 给某一变量赋予数组的某一元素值

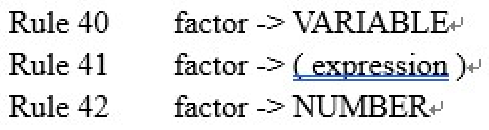
规则:



Rule 29	operation -> VARIABLE = expression↵

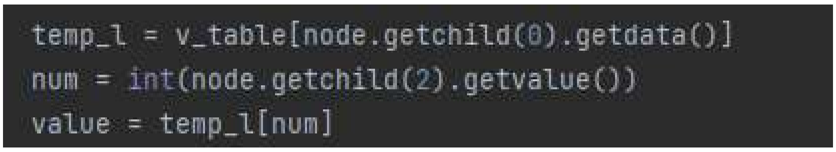


Rule 34	expression -> VARIABLE [factor]↵



Rule 40	factor -> VARIABLE↵
Rule 41	factor -> (expression)↵
Rule 42	factor -> NUMBER↵

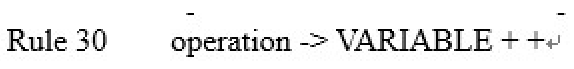
代码:



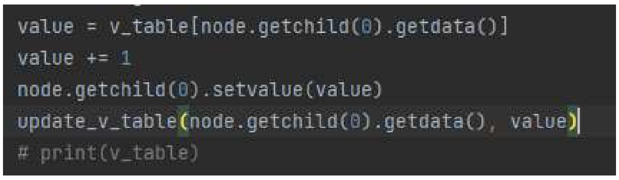
```
temp_l = v_table[node.getchild(0).getdata()]
num = int(node.getchild(2).getvalue())
value = temp_l[num]
```

获取 list 的变量名，获取下标，得到值，然后赋给变量

（4） 自加符号



Rule 30	operation -> VARIABLE + +↵



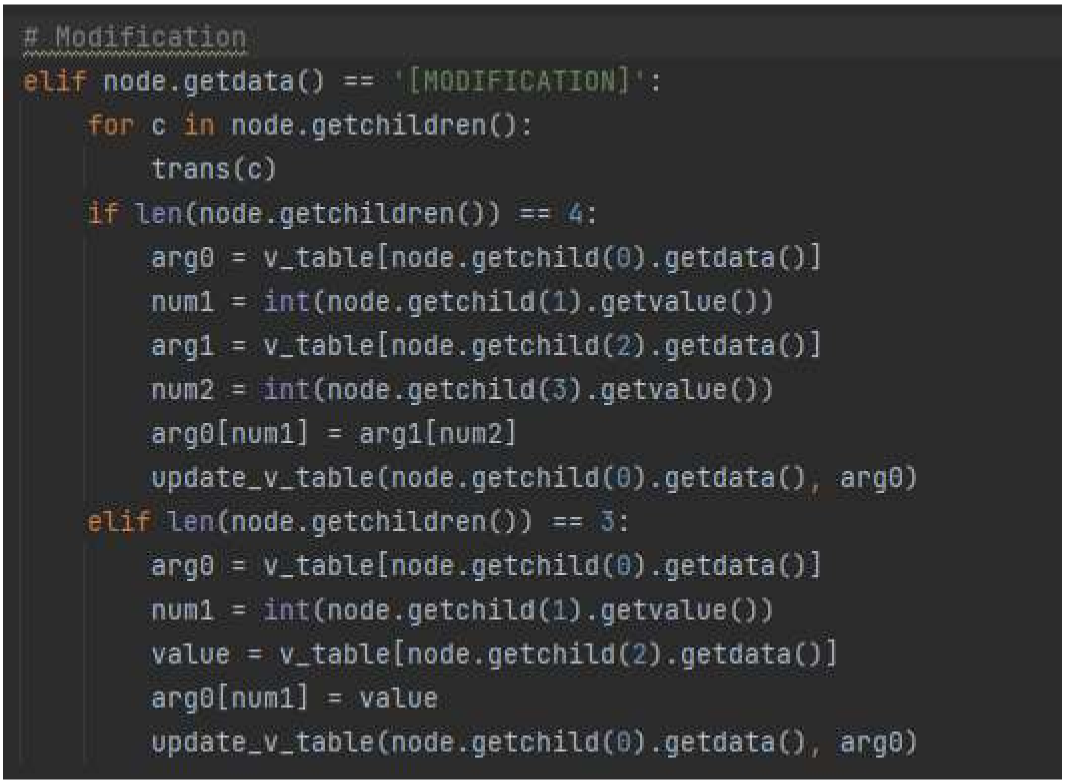
```
value = v_table[node.getchild(0).getdata()]
value += 1
node.getchild(0).setvalue(value)
update_v_table(node.getchild(0).getdata(), value)
# print(v_table)
```

（5） 对数组某一元素值进行修改

规则：

Rule 27 modification -> VARIABLE [factor] = VARIABLE [factor]
Rule 28 modification -> VARIABLE [factor] = VARIABLE

代码：



```
# Modification
elif node.getdata() == '[MODIFICATION]':
    for c in node.getchildren():
        trans(c)
    if len(node.getchildren()) == 4:
        arg0 = v_table[node.getchild(0).getdata()]
        num1 = int(node.getchild(1).getvalue())
        arg1 = v_table[node.getchild(2).getdata()]
        num2 = int(node.getchild(3).getvalue())
        arg0[num1] = arg1[num2]
        update_v_table(node.getchild(0).getdata(), arg0)
    elif len(node.getchildren()) == 3:
        arg0 = v_table[node.getchild(0).getdata()]
        num1 = int(node.getchild(1).getvalue())
        value = v_table[node.getchild(2).getdata()]
        arg0[num1] = value
        update_v_table(node.getchild(0).getdata(), arg0)
```

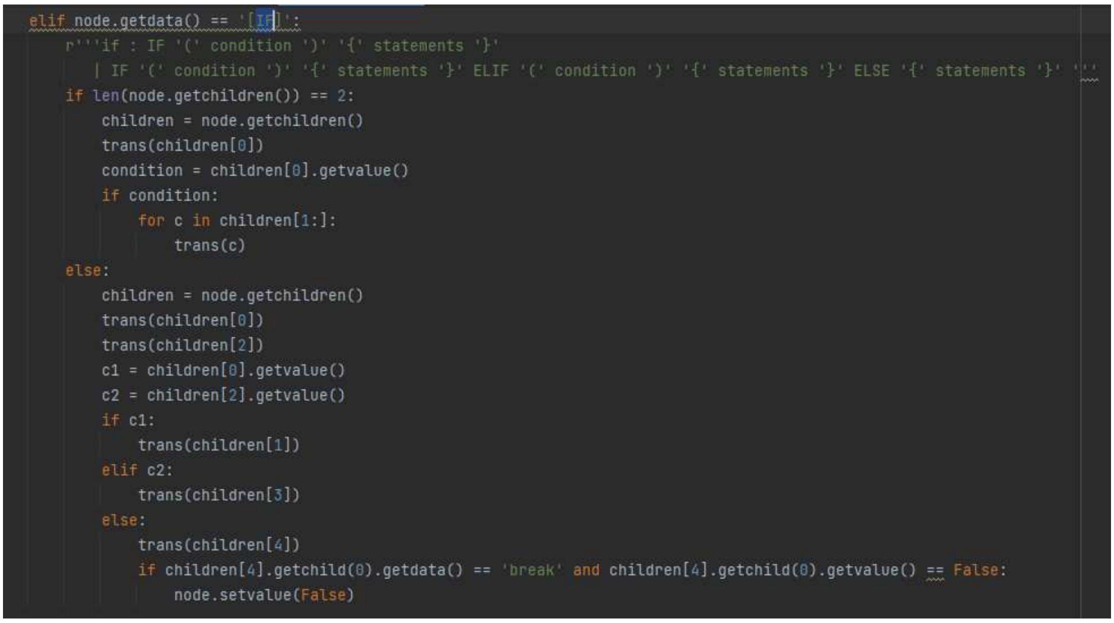
主要依赖的还是子结点的传值和从 v_table 获取变量值

（6） If 语句的解析

规则：

Rule 22 iF -> IF (condition) { statements }
Rule 23 iF -> IF (condition) { statements } ELIF (condition) { statements } ELSE { statements }

代码：



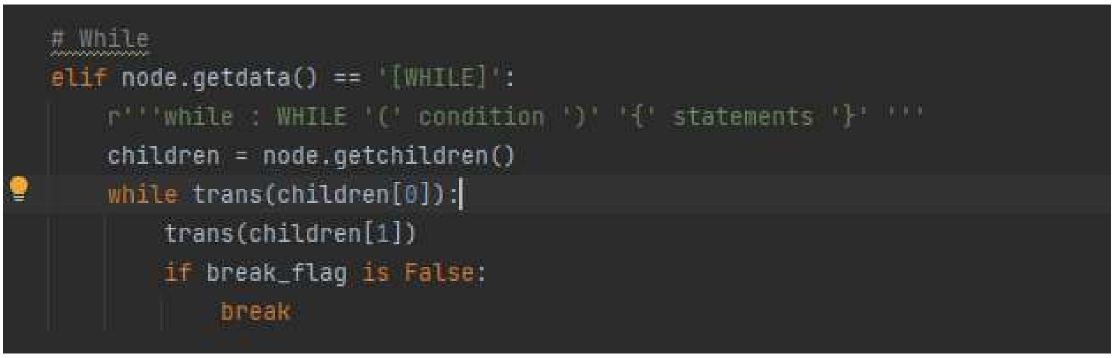
```
elif node.getdata() == '[IF]':
    r'''if : IF '(' condition ')' '{' statements '}'
       | IF '(' condition ')' '{' statements '}' ELIF '(' condition ')' '{' statements '}' ELSE '{' statements '}' '''
    if len(node.getchildren()) == 2:
        children = node.getchildren()
        trans(children[0])
        condition = children[0].getvalue()
        if condition:
            for c in children[1:]:
                trans(c)
    else:
        children = node.getchildren()
        trans(children[0])
        trans(children[2])
        c1 = children[0].getvalue()
        c2 = children[2].getvalue()
        if c1:
            trans(children[1])
        elif c2:
            trans(children[3])
        else:
            trans(children[4])
            if children[4].getchild(0).getdata() == 'break' and children[4].getchild(0).getvalue() == False:
                node.setvalue(False)
```

通过对子结点的 trans 调用, 判断是否满足条件, 决定执行哪一个分支,

并视 break_flag 的情况决定是否要 break

（7） While 语句的解析

Rule 24 whilE -> WHILE (condition) { statements }

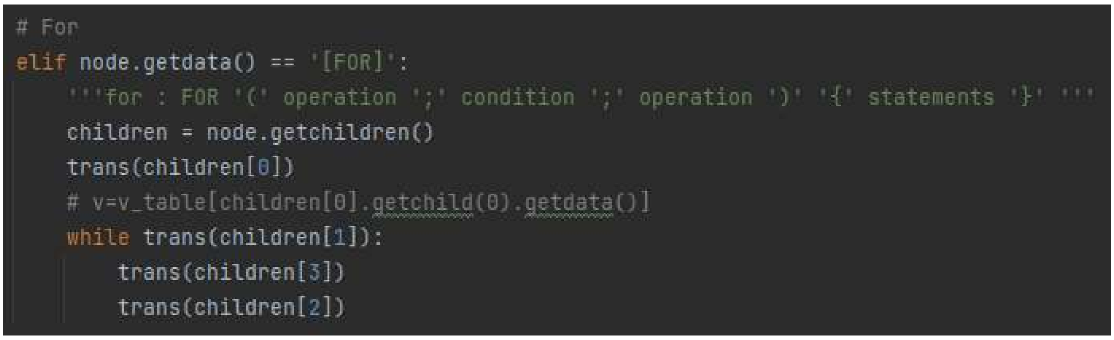


```
# While
elif node.getdata() == '[WHILE]':
    r'''while : WHILE '(' condition ')' '{' statements '}' '''
    children = node.getchildren()
    while trans(children[0]):
        trans(children[1])
        if break_flag is False:
            break
```

子结点不断递归调用， condition 判断是否满足， 看 break_flag 决定是否终止。

（8） For 语句的解析

Rule 14 for -> FOR (operation ; condition ; operation) { statements }



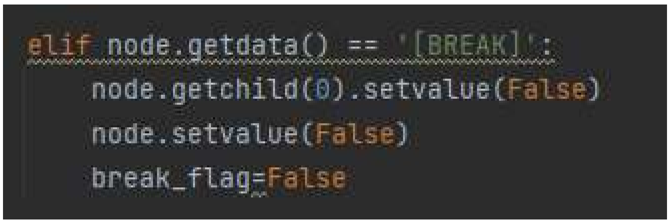
```
# For
elif node.getdata() == '[FOR]':
    '''for : FOR '(' operation ';' condition ';' operation ')' '{' statements '}' '''
    children = node.getchildren()
    trans(children[0])
    # v=v_table[children[0].getchild(0).getdata()]
    while trans(children[1]):
        trans(children[3])
        trans(children[2])
```

按照正常 for 循环时的执行顺序写出对应代码即可，不多作赘述。

（9） Break 语句的解析

设置一个 break_flag，当出现 break 时，就将该值由 True 改为 False，

然后让循环语句去判断是否要 break 即可。



```
elif node.getdata() == '[BREAK]':
    node.getchild(0).setvalue(False)
    node.setvalue(False)
    break_flag=False
```